# TEXTABLE: programmation visuelle pour l'analyse de données textuelles[1]

Aris Xanthos

Université de Lausanne – aris.xanthos@unil.ch

## Abstract

This contribution introduces TEXTABLE, text data analysis open source software offering a visual programming user interface. Implications of the tool's design features as regards interoperability and flexibility will be discussed, as well as its expected adequacy for pedagogical use. A brief introduction to the basics of visual programming for text data analysis is also included.

## Résumé

TEXTABLE est un nouvel outil *open source* de programmation visuelle pour l'analyse de données textuelles. Les implications de la conception de ce logiciel du point de vue de l'interopérabilité et de la flexibilité sont abordées, ainsi que la question que son adéquation pour un usage pédagogique. Une brève introduction aux principes de la programmation visuelle pour l'analyse de données textuelles est également proposée.

**Mots-clés :** Textable, logiciel, *open source*, programmation visuelle, analyse de données textuelles.

## 1. Introduction

L'analyse de données textuelles (ADT) est un domaine scientifique dont le développement semble voué à s'inscrire dans les interactions entre trois types de profils: des chercheurs possédant de fortes compétences en statistique et mathématiques appliquées, à qui incombe la conception et la formalisation des méthodes; des chercheurs en sciences humaines qui, par leur volonté de soumettre les problématiques propres de leurs disciplines à des traitements formels et par leur capacité à mettre en œuvre ce programme, suscitent ou entérinent les innovations méthodologiques; enfin des chercheurs en informatique, qui s'efforcent de rendre les procédures développées par les premiers accessibles aux seconds par l'intermédiaire des outils logiciels. S'il n'est pas rare qu'un même acteur joue plus d'un rôle dans cette distribution, il n'en reste pas moins que le progrès dans ce domaine se joue typiquement sur les trois plans simultanément.

Le rôle de la composante informatique de cette *troïka* a été de plus en plus explicitement thématisé ces dix dernières années, voir notamment (Lamalle et al., 2006 ; Daoust et Marcoux, 2006 ; Heiden, 2006 ; Pincemin, 2008 ; Leblanc, 2010 ; Pincemin et al., 2010). La principale motivation de ce regain d'intérêt semble être une volonté collective de dépasser certaines limitations imposées par la situation des logiciels d'ADT jusqu'au milieu des années 2000. Au risque d'en donner une vision excessivement simplifiée, on peut tenter de résumer cette situation en notant que le « marché » était alors essentiellement occupé par un nombre

1 Le développement du logiciel présenté dans cette contribution a été financé par une subvention du Fonds d'Innovation Pédagogique (FIP) de l'Université de Lausanne. Le logiciel est distribué sous licence open source et librement accessible en suivant les indications données sur la page http://langtech.ch/textable.

restreint de logiciels développés depuis longtemps par les principales équipes de recherche en méthodologie, tels qu'HYPERBASE (Brunet, 2011), LEXICO (Lamalle et al., 2011) ou ALCESTE (Reinert, 1987), pour ne citer que les plus connus. Bien que chacun de ces outils propose des traitements qui lui sont propres, leur communauté de champ d'application justifie que leurs fonctionnalités les plus élémentaires (segmentation, indexation, comptage, etc.) se recoupent largement, ainsi que dans une moindre mesure, certaines de leurs procédures analytiques plus sophistiquées (concordances, cooccurrences, analyse factorielle des correspondances, etc.). En principe, ce sont là des conditions particulièrement favorables à l'interopérabilité entre ces logiciels et pourtant, en raison du développement largement indépendant de chacun d'eux :

> *Les chaînes de traitement se présentent la plupart du temps comme des tunnels méthodologiques qui imposent que les corpus de textes soient mis dans une forme particulière pour pouvoir y pénétrer et qui restituent des résultats dans des formes qui leur sont propres interdisant pratiquement toute comparaison et toute analyse sur la pertinence et l'apport de chacune des étapes du traitement.* (Lamalle et al., 2006:586)

Les conséquences les plus dommageables et absurdes de cet état de fait sont, pour l'utilisateur, la contrainte de jongler entre divers logiciels (et donc divers formats de données et interfaces) sauf à se contenter des capacités de l'un d'entre eux en particulier, et pour le concepteur, le risque de passer plus de temps à reproduire les fonctionnalités les plus populaires des autres logiciels qu'à en développer de nouvelles.

A ce souci d'interopérabilité s'ajoute une préoccupation quant au degré de flexibilité des outils. La question de la segmentation textuelle fournit une bonne illustration de cette problématique. Comme le note André Salem, cité par Bénédicte Pincemin, tout l'édifice de la statistique textuelle repose sur l'hypothèse qu' « il y a des choses... que l'on compte... dans des trucs » (Pincemin, 2008: 950). Or les logiciels cités plus haut adoptent une perspective plutôt rigide sur la nature des choses et des « trucs » : s'ils peuvent être paramétrés quant à la liste de caractères devant être interprétés comme des séparateurs, ou celle des codes permettant d'identifier des parties de textes, ils n'admettent généralement que de modestes dérogations aux principes que les choses sont des mots graphiques (ou des propriétés qui leur sont associées, telles que lemme ou catégorie morphosyntaxique), les « trucs » sont des textes ou des parties de textes (paragraphes, phrases, etc.), et les unes ne sont certainement pas assimilables aux autres :

> [..] *les logiciels textométriques réalisent simultanément deux opérations distinctes. La première que nous appelons* microsegmentation *consiste à segmenter la chaîne textuelle en* occurrences *(angl.* token*), puis à regrouper ces dernières, à partir de critères d'identification variables, en des ensembles d'occurrences considérées comme identiques que l'on appellera des* types *(angl.* types*). La seconde opération que nous appelons* macrosegmentation *permet de repérer les frontières des parties ou fragments du corpus que l'on désire comparer.* (Lamalle et al., 2006: 586)

On peut toutefois se demander dans quelle mesure cette distinction entre deux types d'opérations de segmentation est utile d'un point de vue conceptuel et applicatif. Pour prendre un exemple simple et concret, est-il souhaitable de penser et d'effectuer le calcul du nombre moyen de mots par phrase de façon radicalement différente du calcul du nombre moyen de phrases par texte ou de lettres par mot ? Ne semble-t-il pas à la fois plus simple et plus productif de les envisager comme les réalisations d'un même processus variant quant à ce qui

joue le rôle de chose ou de truc ? C'est du moins la position soutenue par Bénédicte Pincemin :

> [..] *parties et unités se comportent comme des rôles, de contenant ou de contenu, dévolus à des segments textuels. Autrement dit, il n'y a pas d'unités ou de parties par nature, mais par fonction, et ce qui est partie (contenant) dans une analyse peut devenir unité (contenu) dans une autre, par un simple changement de granularité de l'analyse. Segmentation en unités typées, et délimitation d'une partition, sont alors simplement deux manières de voir une même réalité.* (Pincemin, 2008:955)

La question de l'ergonomie des logiciels d'ADT a également reçu une attention particulière dans un article récent de Jean-Marc Leblanc (2010). Celui-ci note que si ces outils sont avant tout destinés à un public de chercheurs et donc orientés vers la mise en œuvre d'une démarche dont la connaissance et la compréhension sont présupposées, il n'en est pas moins souhaitable « de rendre meilleures nos interfaces, et de les adapter à un public auquel il convient de penser également, ceux qui se forment à ces outils » (Leblanc, 2010:1058). C'est précisément la perspective d'un usage pédagogique qui a motivé le développement de TEXTABLE, le logiciel *open source* qui fait l'objet de cette contribution. La section suivante montrera comment sa conception s'efforce de répondre aux préoccupations évoquées dans cette introduction. La section 3 proposera une brève introduction aux principes de la programmation visuelle pour l'ADT avec TEXTABLE. En conclusion, nous reviendrons en particulier sur le bilan de la première utilisation du logiciel dans un contexte pédagogique en vue de formuler des propositions pour son développement futur.

## 2. Design

La conception de TEXTABLE a été principalement guidée par le souci de prendre en compte les considérations mentionnées dans la section précédente. Il a donc été établi que le logiciel devait satisfaire en premier lieu les critères suivants :

- pouvoir servir de support à l'enseignement des concepts et méthodes propres de l'ADT à des utilisateurs non spécialistes – typiquement des étudiants ou chercheurs en sciences humaines ;
- offrir une flexibilité élevée du point de vue des traitements possibles, en particulier en ce qui concerne la segmentation textuelle ;
- favoriser l'interopérabilité aussi bien en ce qui concerne les formats d'entrée que de sortie.

Les choix de conception qui ont été effectués en vue de trouver le meilleur compromis entre ces contraintes sont relativement différents de ceux qui sous-tendent les logiciels d'ADT classiques ou plus récents comme IRaMuTeQ (Ratinaud, 2009), TXM (Heiden et al., 2010) ou Le Trameur (Fleury, 2013). A ce titre, la particularité la plus remarquable de TEXTABLE est sans doute l'adoption d'une interface de *programmation visuelle*, de type *flux de données* (*dataflow)* en particulier.[2] Dans ce contexte, l'utilisateur est amené à composer un *schéma*, soit une chaîne de traitement, en arrangeant graphiquement des instances de *widgets*

[2] L'interface en question est celle d'Orange Canvas (Demsar et Zupan, 2004), logiciel d'analyse de données open source dont TEXTABLE est une extension. Elle présente des similarités évidentes avec celle du produit commercial Coheris Spad (http://www.coheris.fr/fr/page/produits/Spad.html), en particulier ses « diagrammes » (anciennement « filières »).

représentant des fonctionnalités spécifiques (telles qu'importation, recodage, segmentation, etc.) et en les reliant par des connexions qui déterminent le cheminement des données dans le système (voir la figure 1 ci-dessous).

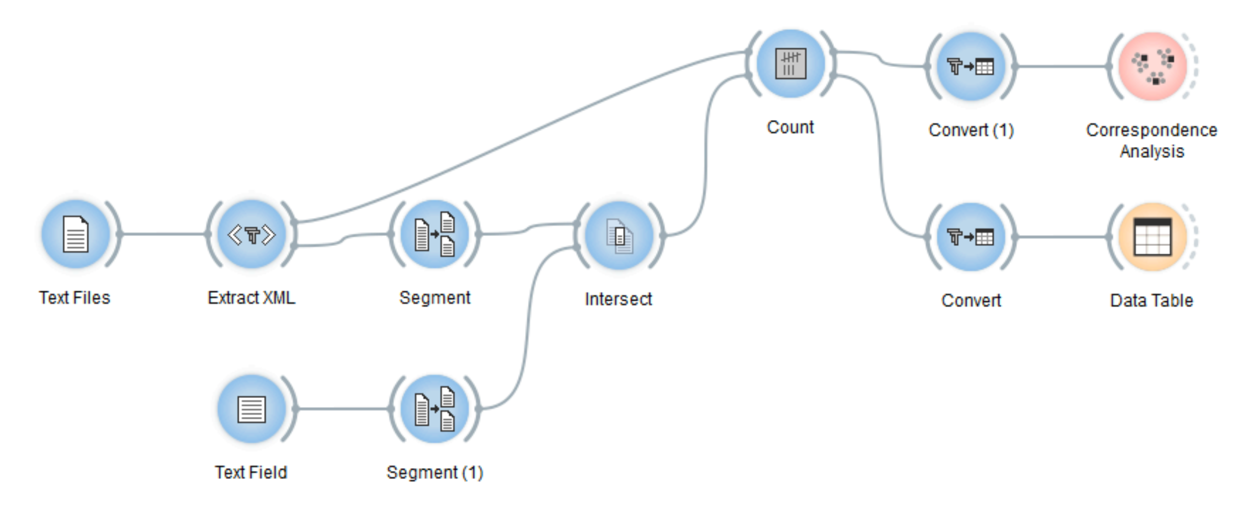



*Figure 1. Exemple de schéma TEXTABLE*

Ce design a des conséquences importantes du point de vue pédagogique, à commencer par le fait que rien n'est « gratuit » pour l'utilisateur : l'usage le plus trivial, comme la visualisation du contenu d'un fichier par exemple, implique de créer deux instances de *widgets* et une connexion entre eux ; une application plus réaliste comprendra facilement une dizaine d'instances ou plus, qu'il s'agira de connecter et configurer correctement. De façon peut-être surprenante, cet état de fait est envisagé ici comme un avantage dans une perspective didactique. Pour reprendre l'argumentation métaphorique de (Resnick et al., 1996) :

> *Would you rather that your children learn to play the piano, or learn to play the stereo?... In the field of educational technology, there has been too much emphasis on the equivalent of stereos and CDs, and not enough emphasis on... "computational construction kits" that, like pianos, enable people to express themselves in ever-more complex ways, deepening their relationships with new domains of knowledge.* (Resnick et al., 1996:41-42)

De la même façon, TEXTABLE peut être envisagé comme un kit de construction pour l'ADT, dont la vocation est de stimuler l'apprentissage en donnant un rôle actif et créatif à l'utilisateur. Cela ne revient pas à dire qu'il est facile d'apprendre à utiliser le logiciel, mais plutôt que l'effort requis pour gravir sa courbe d'apprentissage est consenti d'autant plus facilement que l'utilisateur se sent impliqué dans une activité de construction.

L'adoption d'une interface de type flux de données confère également à TEXTABLE une flexibilité considérable. Les 17 types de *widgets* mis à disposition de l'utilisateur permettent une variété de constructions distinctes, dont certaines sont d'ailleurs équivalentes du point de vue de leurs résultats. Cette versatilité est encore renforcée par une autre décision de conception fondamentale, soit l'usage très systématique des expressions régulières pour la configuration des instances de *widgets*. Ainsi, le *widget Segment* a pour fonction de créer une segmentation en identifiant, dans les données textuelles qu'il reçoit en entrée, des segments dont la forme correspond à une expression régulière donnée ; dans le cas représenté sur la figure 2 ci-dessous, par exemple, il s'agit des mots graphiques, soit les séquences de caractères alphanumériques contigus les plus longues possibles, ce qui équivaut à la segmentation obtenue à partir d'une liste de délimiteurs dans LEXICO 3 par exemple. Pour

prendre la mesure de la souplesse offerte par cette interface, on peut observer que de très petites modifications de l'expression régulière aboutissent à des résultats complètement différents: *\w* pour une segmentation en caractères alphanumériques ou .+ pour une segmentation en lignes, pour ne citer que les cas les plus élémentaires.

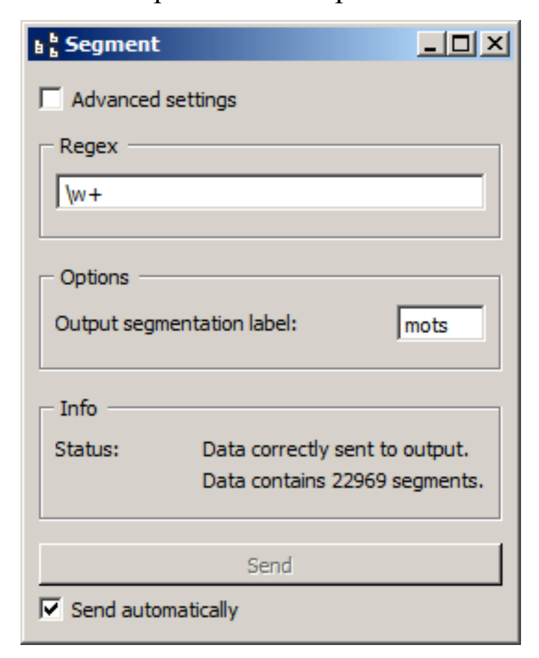



*Figure 2. Interface du widget* Segment

Considérant que la segmentation (typiquement en mots) est l'une des opérations les plus communément effectuées dans le cadre de l'ADT, on s'étonnera sans doute de constater que TEXTABLE n'offre pas de façon plus simple de l'effectuer. Il s'ensuit qu'à moins de disposer de connaissances préalables concernant la syntaxe des expressions régulières, l'utilisateur doit passer par une phase d'apprentissage pour pouvoir effectuer la moindre analyse de données textuelles. Une fois encore, il s'agit là d'un parti pris pédagogique, et si le lecteur le juge problématique, il sera peut-être rassuré de noter que les trois expressions mentionnées plus haut (*\w+*, *\w* et .+) suffisent à couvrir les besoins élémentaires d'une majorité d'utilisateurs, d'une part, et que trois instances de *widgets* judicieusement arrangées suffisent à construire un environnement interactif simple pour apprendre l'utilisation des expressions régulières.

Parmi les choix de conception ayant vocation à favoriser l'interopérabilité de TEXTABLE, le plus radical est le suivant : comme son nom le suggère, le logiciel prend en entrée des données en format texte brut et produit en sortie des tableaux de données (exportables sous forme de texte délimité par des tabulations)[3]. En ce qui concerne les entrées, on a évité en particulier d'adopter un format *ad hoc*[4] pour la délimitation et l'annotation de parties de texte (comme c'est le cas dans LEXICO 3 ou IRaMuTeQ par exemple), dans la mesure où le mécanisme de segmentation à base d'expressions régulières mentionné plus haut permet d'émuler cette fonctionnalité avec plus de flexibilité. La décision de limiter les sorties du logiciel à des tableaux de données a été motivée en premier lieu par une volonté d'éviter de dupliquer des traitements effectués de façon tout à fait satisfaisante par de nombreux logiciels d'analyse de données tabulées tels que l'excellent projet *open source* R (R Development Core

[3] Pour être complet, ajoutons qu'il est possible d'exporter des versions recodées des données textuelles d'entrée.

[4] Le logiciel offre toutefois la possibilité d'exploiter des informations encodées selon le format *standard* XML.

Team, 2008). Il faut toutefois préciser qu'Orange Canvas, dont TEXTABLE est une extension, est lui-même un logiciel d'analyse de données généraliste et donc que de nombreux traitements analytiques sophistiqués (analyses factorielles, classification supervisée ou non, régression multiple, etc.) sont directement accessibles sans devoir exporter de données vers un logiciel tiers.

## 3. Principes de construction de schémas pour l'ADT

Après ce tour d'horizon des principales particularités du *design* de TEXTABLE, cette section se propose d'introduire les bases de la programmation visuelle pour l'ADT sur la base d'un exemple graduellement complexifié.

### *3.1. Importation et visualisation de données textuelles*

Le schéma représenté sur la figure 3 ci-dessous illustre plusieurs aspects fondamentaux du fonctionnement de TEXTABLE. Il contient deux instances de *widgets* (*Text Field* et *Display*) reliés par une connexion allant de la sortie du premier (sur son côté droit, par convention) vers l'entrée du second (sur son côté gauche).

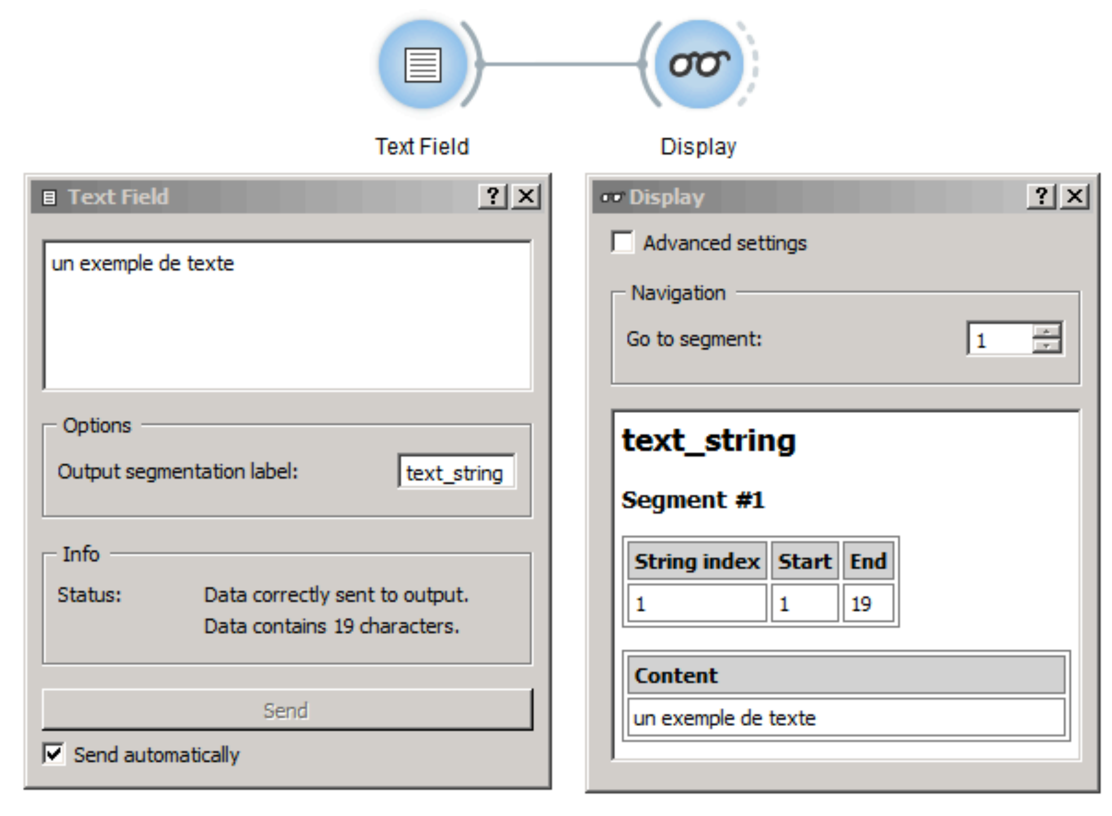



*Figure 3. Saisie au clavier et visualisation de données textuelles*

*Text Field* est l'un des trois *widgets* permettant d'importer des données textuelles dans le système (les deux autres, *Text Files* et *URLs*, servent respectivement à lire le contenu de fichiers et télécharger des données en ligne). La vocation de ce *widget* est surtout pédagogique: il permet de saisir une chaîne de caractères au clavier et de lui associer une *segmentation*, au sens spécifique de ce terme dans TEXTABLE, soit une liste de *segments* définis par leur position de départ et de fin dans la chaîne. Par défaut, les 19 caractères de la chaîne *un exemple de texte* sont considérés comme faisant partie d'un segment unique. La

segmentation contenant cet unique segment se voit en outre affecter un *label*, soit une chaîne permettant de l'identifier dans la suite des traitements (*text_string* en l'occurrence).

La segmentation ainsi créée est émise par *Text Field* en direction de *Display*, dont la fonction première est d'afficher la représentation interne d'une segmentation. On voit bien ainsi que la chaîne entière est traitée comme un segment unique du caractère 1 au caractère 19 inclus ; pour fixer la terminologie, on distinguera l'*adresse* du segment (ses positions de départ et de fin dans une chaîne donnée) de son *contenu*, soit la sous-chaîne correspondant à cette adresse dans la chaîne originale. Notons que lorsque la case *Send automatically* est cochée dans l'interface de *Text Field*, toute modification apportée à la chaîne de caractères est directement reportée sur l'affichage du *widget Display*.

### *3.2. Fusion de segmentations*

TEXTABLE permet de combiner facilement plusieurs segmentations pour en former une seule au moyen du *widget Merge*. Celui-ci fait partie des *widgets* qui peuvent recevoir plusieurs connexions entrantes simultanées, contrairement à *Display* par exemple. Dans l'exemple de la figure 4 ci-dessous, une seconde instance de *Text Field* a été créée (avec la chaîne *un autre texte* et le label *text_string2*), et les deux sources ont été connectées à une instance de *Merge*, elle-même connectée à *Display*.

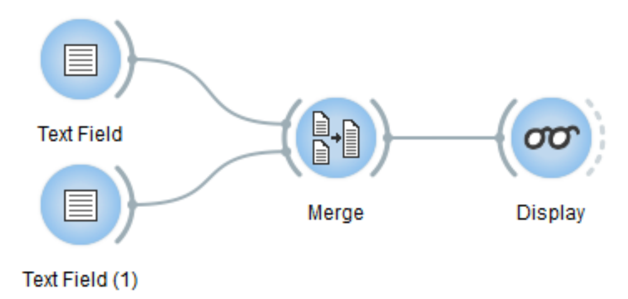



*Figure 4. Fusion de segmentations*

La configuration de *Merge* et le résultat affiché dans *Display* sont visibles sur la figure 5 ci-dessous. L'interface de *Merge* permet de sélectionner l'ordre dans lequel les deux segmentations entrantes sont concaténées ainsi que le label affecté à la segmentation sortante. Une autre fonctionnalité importante est la possibilité d'associer aux segments sortants des *annotations* basées sur les labels des segmentations entrantes (case à cocher *Import labels with key*). Dans le cadre de TEXTABLE, les annotations sont définies comme des paires clés – valeurs associées à des segments ; en l'occurrence, la clé est la chaîne *composant* et la valeur est le label de chaque segmentation entrante. Les annotations associées à chaque segment apparaissent entre son adresse et son contenu dans l'affichage de *Display*.

Notons en passant que toutes les chaînes de caractères sont converties en Unicode en interne, si bien qu'une segmentation composite peut tout à fait contenir des données provenant de plusieurs encodages différents.

### *3.3. Matrice documents–termes*

La matrice documents–termes (ou trucs–choses) est peut-être le type de table le plus communément employé en ADT. Le *widget Count* de TEXTABLE permet de produire de telles tables (entre autres types de tables de fréquences). Sur la figure 6 ci-dessous, on peut noter tout d'abord l'insertion d'une instance de *Segment* à la sortie de *Merge* ; cette instance, configurée de la même façon que celle représentée sur la figure 2 ci-dessus, produit une

segmentation en mots graphiques à partir de la segmentation *sources* et affecte le label *mots* à cette nouvelle segmentation.

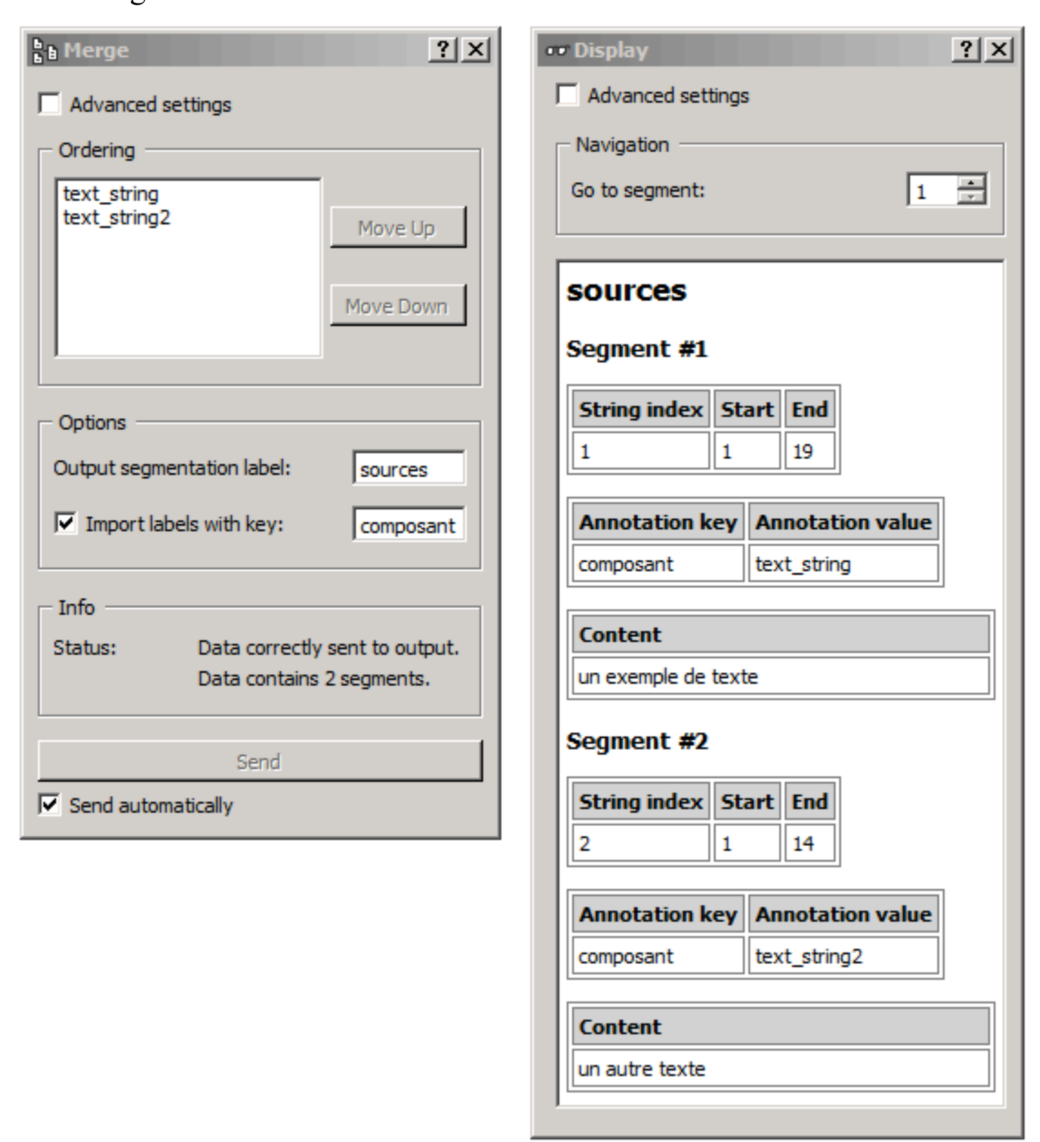



*Figure 5. Fusion de segmentations (widget* Merge*)*

Les deux segmentations (*sources* et *mots*) sont alors envoyées au *widget Count*. Il est important de noter que *Count* est l'un des cinq *widgets* prenant en entrée une ou plusieurs segmentations et produisant en sortie une *table* (et non une segmentation comme tous les *widgets* considérés jusqu'ici).[5] Comme le montre la figure 6 ci-dessous, l'interface de ce *widget* permet de définir la nature des choses ou *unités* (*Units*) et des trucs ou *contextes* (*Contexts*) : en l'occurrence, les unités sont les segments de la segmentation *mots* et les contextes ceux de la segmentation *sources*; on indique par ailleurs que les types de contextes[6]

---

[5] Les quatre autres widgets de ce type sont *Context*, qui permet de produire des concordances ou listes de collocations, *Length*, spécialisé dans les calculs de longueur (*cf.* section 3.4 ci-dessous), *Variety*, qui permet d'évaluer la diversité des segments, et *Annotation*, dont la fonction est d'extraire les annotations associées aux segments (par exemple en vue d'une application de classification supervisée).

[6] Une conséquence importante de l'abolition de la distinction entre micro- et macro-segmentation (au sens de Lamalle et al., 2006:586) est que les contextes sont regroupés en types au même titre que les unités.

sont définis sur la base des valeurs associées à la clé d'annotation *composant* (*Annotation key: composant*) plutôt que du contenu des segments. Après un passage par le *widget Convert*, qui ne sert ici qu'à transposer la matrice documents–termes pour le plaisir des yeux, le résultat peut être visualisé au moyen du *widget Data Table* (*widget* standard d'Orange Canvas) comme sur la figure 6 ci-dessous.

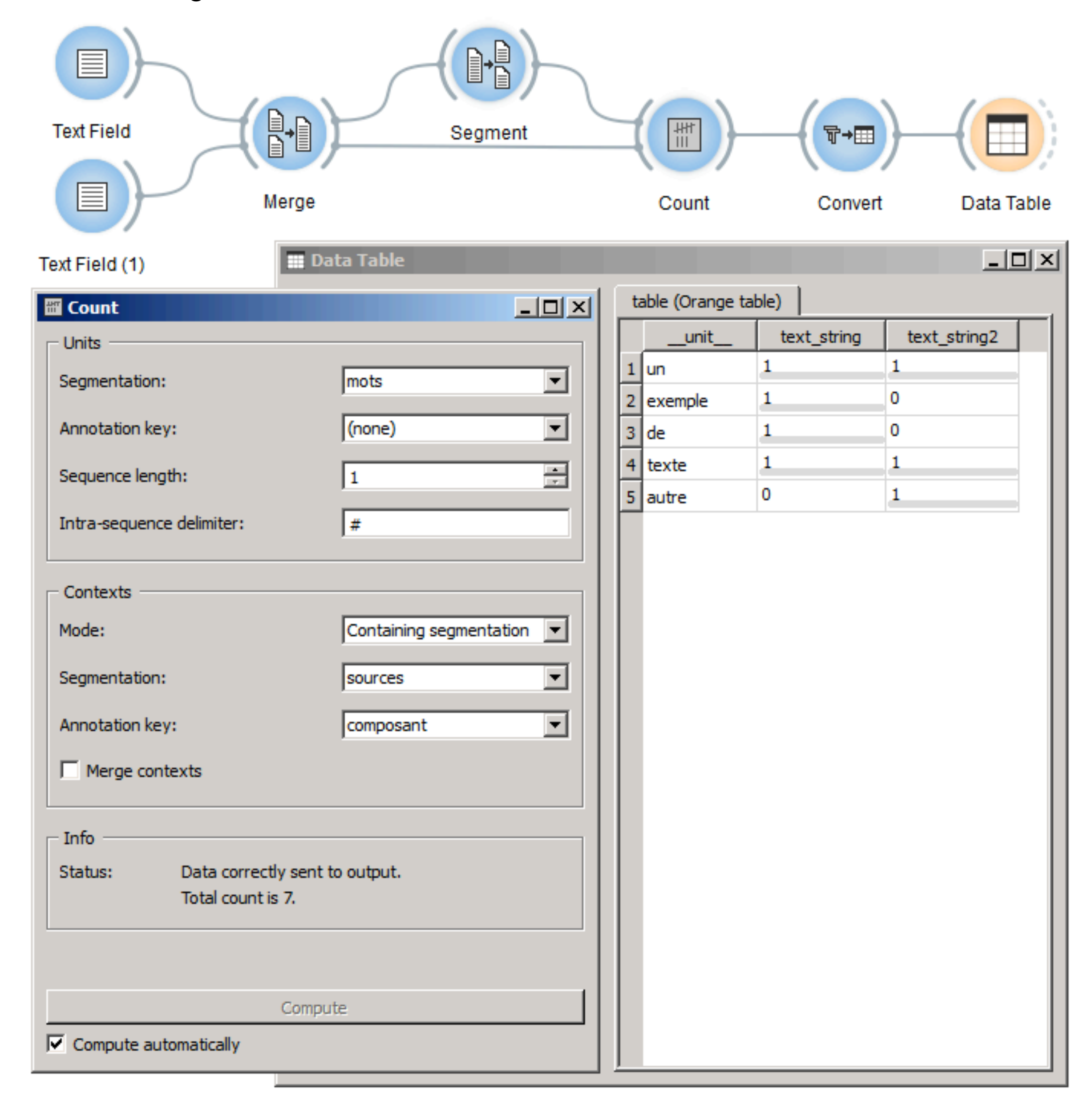



*Figure 6. Matrice documents–termes (widgets* Count *et* Data Table*)*

### *3.4. Calcul de longueur moyenne*

Pour illustrer la flexibilité du logiciel, la dernière partie de cet exemple sera consacrée à l'exposé des modifications à apporter au schéma précédent pour calculer plutôt la longueur moyenne des mots (en nombre de lettres) dans les deux chaînes de caractères. Notons dès l'abord que ce calcul implique nécessairement trois segmentations distinctes : outre les segmentations en chaînes (*sources*) et en mots (*mots*) déjà produites, il faut encore construire une segmentation en lettres. Celle-ci s'obtient aisément en insérant une nouvelle instance de

*Segment* à la sortie de la précédente (voir figure 7 ci-dessous) et en la configurant avec l'expression régulière \w (*cf.* section 2 ci-dessus); on lui affectera par ailleurs le label *lettres*.

Il faut également remplacer l'instance du *widget Count* par *Length*. Les trois segmentations (*sources*, *mots* et *lettres*) sont envoyées à ce nouveau *widget*, dont la configuration apparaît sur la figure 7 ci-dessous. Les contextes sont définis comme précédemment (*Segmentation: source* et *Annotation key: composant*), mais cette fois-ci c'est la segmentation *lettres* qui spécifie les unités ; enfin la segmentation *mots* définit les éléments intermédiaires sur lesquels la moyennisation doit être effectuée. La table résultante montre que si le nombre de mots varie entre les deux chaînes, leur longueur moyenne se trouve être égale à 4 dans les deux cas.

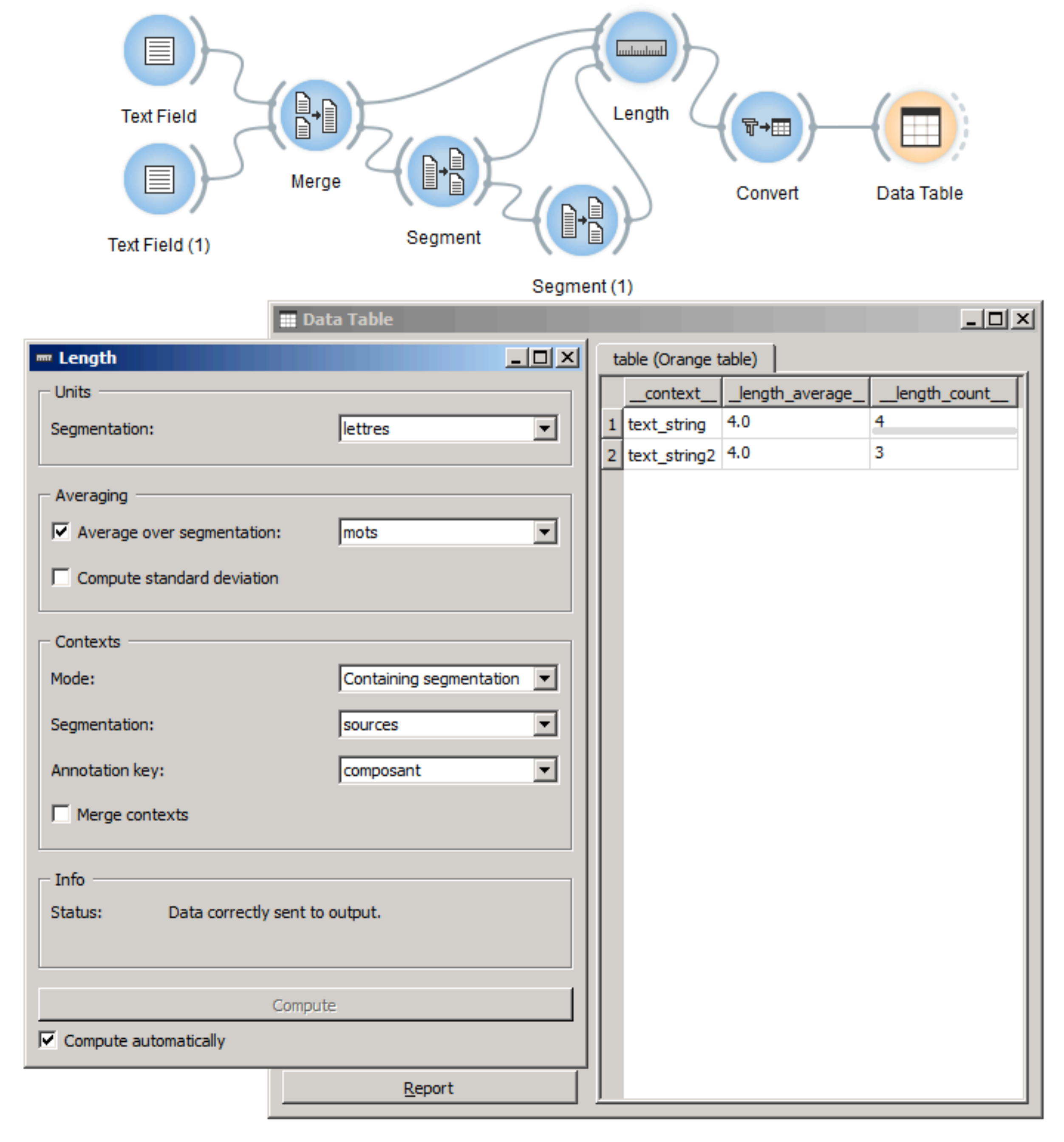



*Figure 7. Nombre moyen de lettres par mot (widget* Length*)*

# 4. Discussion et conclusion

Cette contribution s'est donné pour objectifs principaux de rendre compte des principales considérations qui ont guidé le développement de TEXTABLE et de présenter les bases de la programmation visuelle pour l'ADT au moyen de ce nouvel outil. Dans le but de favoriser le caractère didactique de la présentation, seule une petite partie des 17 *widgets* du logiciel ont été discutés, et bon nombre de leurs fonctionnalités sont restées dans l'ombre. Par ailleurs, comme un lecteur attentif l'aura peut-être noté, la plupart des *widgets* offrent le choix entre une interface simple ou avancée[7] (case à cocher : *Advanced settings*). A titre d'exemple, la version avancée de *Segment* permet de définir plusieurs expressions régulières et de les ordonner ; d'indiquer si elles caractérisent la forme des segments visés (c'est le cas par défaut) ou celle des délimiteurs qui les séparent ; de capturer des portions du contenu des segments et les utiliser pour générer dynamiquement des clés et/ou des valeurs d'annotation etc. En somme, la simplicité des exemples présentés ici ne doit pas laisser croire que les capacités du logiciel sont insuffisantes pour aborder de vraies questions de recherche appliquées en ADT.

Dans la perspective d'une application concrète, l'une des plus importantes limitations de TEXTABLE est le temps d'exécution requis pour manipuler de grandes segmentations, soit contenant des milliers de segments – dont la nature importe moins que le nombre (segmenter un texte en mots est toujours plus rapide qu'en lettres, par exemple). On peut toutefois limiter l'impact de ce problème en se conformant au principe général de subdiviser les tâches les plus lourdes : ainsi, produire une segmentation en lignes, puis la *sous*-segmenter en mots et enfin en lettres est spectaculairement plus rapide que de produire directement la segmentation la plus fine. Quoi qu'il en soit, la facilité avec laquelle le logiciel permet d'effectuer des tâches pour lesquelles la seule alternative est souvent le recours à un langage de programmation général comme Perl ou Python amène à relativiser l'obstacle que peut constituer le temps d'exécution de certains calculs. Le logiciel a déjà fait l'objet d'une première utilisation dans le cadre d'un enseignement annuel d'initiation à l'ADT pour des étudiants de baccalauréat universitaire en sciences humaines. L'expérience suggère que l'usage du logiciel (d'abord dans des travaux dirigés, puis en autonomie dans le cadre de projets personnels ou collectifs) favorise effectivement la compréhension des méthodes et celle des concepts qui les sous-tendent, notamment grâce à la possibilité de visualiser et manipuler symboliquement les données textuelles, les unités de traitement et leur arrangement, enfin les résultats de l'application des unes aux autres. Par ailleurs, les étudiants peuvent être relativement ambitieux dans la planification de leurs projets, dans la mesure où l'outil leur permet d'effectuer de nombreuses tâches qui nécessiteraient autrement des compétences en programmation dont ils ne disposent pas à ce stade de leurs études. Dans leur retour concernant l'usage de TEXTABLE pour la réalisation desdits projets, ils évoquent typiquement des débuts un peu laborieux, puis un « déclic » à la suite duquel l'utilisation du logiciel devient à la fois ludique et productive.

L'une des meilleures surprises apparues lors de cette première expérience pédagogique avec l'outil est l'intérêt de son utilisation comme support de communication. Qu'il s'agisse pour l'enseignant de présenter un élément de méthodologie ou pour les étudiants d'exposer le travail effectué dans le cadre de leurs projets, les schémas TEXTABLE s'avèrent très efficaces grâce à leur caractère visuel et aux possibilités qu'ils offrent en terme d'interactivité. Le degré

[7] Il faut y voir une manifestation du souci pédagogique qui a guidé le développement du logiciel.

de maîtrise des expressions régulières est sans doute le facteur conditionnant le plus fortement la capacité des utilisateurs à exploiter les possibilités du logiciel. S'il ne paraît pas insurmontable d'associer les expressions .+, \w+ et \w aux processus de segmentation en lignes, mots et lettres respectivement, il faut une familiarité considérable avec ces conventions syntaxiques pour produire une expression identifiant, mettons, des mots comprenant 3 à 7 lettres et n'étant pas précédés par une chaîne donnée. Le développement de compétences spécifiques en matière de rédaction d'expressions régulières apparaît donc comme un enjeu fondamental pour l'enseignement des méthodes et concepts de l'ADT avec TEXTABLE. Dans la mesure où le logiciel lui-même permet de mettre en place un environnement pour cet apprentissage, il semble opportun de faire de la conception d'activités pédagogiques à cet effet une priorité pour son développement futur.

## Références

Brunet E. (2011). HYPERBASE© : Logiciel hypertexte pour le traitement documentaire et statistique des corpus textuels – Manuel de référence. (ftp://ancilla.unice.fr/manuel.pdf)

Daoust F. et Marcoux Y. (2006). Logiciels d'analyse textuelle: vers un format XML-TEI pour l'échange de corpus annotés. In *Actes des 8èmes Journées internationales d'Analyse statistique des données Textuelles (JADT'2006)*, pp. 527-340.

Demsar J. et Zupan B. (2004). *Orange: From Experimental Machine Learning to Interactive Data Mining, White Paper*. Faculty of Computer and Information Science, University of Ljubljana. (http://orange.biolab.si/wp/orange.pdf)

Fleury S. (2013). Le Métier Textométrique aka Le Trameur – Manuel d'utilisation. (http://www.tal.univ-paris3.fr/trameur/leMetierLexicometrique.pdf)

Heiden S. (2006). Un modèle de données pour la textométrie: contribution à une interopérabilité entre outils. In *Actes des 8èmes Journées internationales d'Analyse statistique des Données Textuelles (JADT'2006)*, pp. 487-498.

Heiden S., Magué J-P. et Pincemin B. (2010). TXM: Une plateforme logicielle open-source pour la textométrie – conception et développement. In *Actes des 10èmes Journées d'Analyse statistique des Données Textuelles (JADT'2010)*, pp. 1021-1032.

Lamalle C., Fleury S. et Salem A. (2006). Vers une description formelle des traitements textométriques. In *Actes des 8èmes Journées internationales d'Analyse statistique des données Textuelles (JADT'2006)*, pp. 583-593.

Lamalle C., Martinez W., Fleury S. et Salem A. (2003). LEXICO 3: Outils de statistique textuelle – Manuel d'utilisation. (http://www.tal.univ-paris3.fr/lexico/manuelsL3/manuel-3.41.pdf)

Leblanc J.-M. (2010). Nouvelles fonctionnalités pour la visualisation des données textuelles et de résultats : Pour une approche plus ergonomique des dispositifs lexicométriques. In *Actes des 10èmes Journées d'Analyse statistique des Données Textuelles (JADT'2010)*, pp. 1057-1068.

Pincemin B., Heiden S., Lay M.-H., Leblanc J.-M. et Viprey J.-M. (2010). Fonctionnalités textométriques: proposition de typologie selon un point de vue utilisateur. In *Actes des 10èmes Journées d'Analyse statistique des Données Textuelles (JADT'2010)*, pp. 341-353.

Pincemin B. (2008). Modélisation textométrique des textes. In *Actes des 9èmes Journées d'Analyse statistique des Données Textuelles (JADT'2008)*, pp. 949-960.

R Development Core Team. (2008). *R: A language and environment for statistical computing*. Vienne (Autriche): R Foundation for Statistical Computing. (http://www.R-project.org)

Ratinaud P. (2009). *IRaMuTeQ: Interface de R pour les Analyses Multidimensionnelles de Textes et de Questionnaires*. (http://www.iramuteq.org)

Reinert M. (1987). Un logiciel d'analyse lexicale (ALCESTE). *Cahiers Analyse des Données,* 4: 471-484.

Resnick M., Bruckman A. et Martin F. (1996). Pianos not stereos: creating computational construction kits. *Interactions*, 3(5): 40-50.